REMARKS/ARGUMENTS

Claims 1-4 have been amended. Claim 5 has been canceled. Claim 6 has been canceled. Claim 7 has been canceled. Claim 8 has been canceled. Claim 9 has been amended.

The examiner has acknowledged that claims 1 – 4 and claim 9 have now been amended to correct editorial errors and clear up any matters of form. Claims 5 – 8 have been canceled as being drawn to an embodiment no longer of interest to applicant.

Claim 1 has been amended for the following reasons:
To clear up any matters of form.

Claim 2 has been amended for the following reasons:
To clear up any matters of form.

Claim 3 has been amended for the following reasons:
To clear up any matters of form.

Claim 4 has been amended for the following reasons:
To clear up any matters of form.

Claim 9 has been amended for the following reasons:
To clear up any matters of form.

Attached hereto is a marked-up version of the changes made to the specification, claims and drawings by the current amendment. The attached page is captioned **"Version with markings to show changes made."**

Applicant respectfully requests that a timely Notice of Allowance be issued in this case.

Respectfully submitted,

INVENTOR

BY Gary John Corey

Gary John Corey
CamSoft Corporation
(951) 674-8100

In the Drawings:

The content of FIG 1 has been amended. The page numbers for FIG 1 through FIG 9 have been amended as a result of the addition of FIG 10.

VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the Specification:

A new drawing description has been added to the section entitled "BRIEF DESCRIPTIONS OF THE SEVERAL VIEWS OF THE DRAWINGS ", after FIG 9, as indicated in the clean copy and shown below:

FIG 10. is a flowchart in block diagram form of each technology element.

The section entitled "DETAILED DESCRIPTION OF THE INVENTION" has new rewritten content for paragraph Nos. [0022], [0023], [0029] and [0030], as indicated in the clean copy and shown below. Paragraph Nos. [0001] through [0012], [0015] through [0021] and [0024] through [0028] remain as is in the Specification. Paragraph Nos. [0013] and [0014] have been deleted. Paragraph Nos. [0031] through [0053] have been added.

[0013] ~~On FIG 1. the group of boxes under the section OPTIONAL SETTINGS is described as follows:~~

[0014] ~~This is an array of miscellaneous settings that do not directly relate to features needed for multiple-axes tool comp. They are presented here to fully disclose how multiple-axes tool comp interacts with the various other operator settings and choices.~~

~~DRY RUN is used to switch on or off a mode in which the Z axis, Spindle and Feed mode will be disabled or not. Commonly used in test runs.~~

~~BITMAP G CODE DISPLAY will refresh the GCODE window whenever another window pops up over it.~~

~~GRAPHICS: SOLID VS WIREFRAME switches the graphics display to either a solid model or wireframe.~~

~~TOLERANCE a user-provided value in which to perform the calculations.~~

~~BLOCK SKIP CHARACTER is a user-defined character to tell the computer to skip this line of data or not.~~

~~TEACH FILE NAME is a file name in which to store all of the locations used.~~

~~SOLID STOCK Begin Z @ specifies the Z axis beginning of the solid model. Typically 0.~~

~~SOLID STOCK Extra Stock adds extra material or stock around the edges or diameter.~~

~~FANUC ARC CENTER to be calculated from Absolute, incremental or radii given arc centers.~~

[0022] Shown below are the variables used and how the calculations are made in the central set of math routine algorithms.

Element Title: Vector and Matrix Subroutine

U,V,W are the end result of the compensated tool positions.
D = the distance or combined length of FIG 2. Dim "A" Item 2, Dim "B" Item 3 and Dim "C" Item 4.
Vx,Vy,Vz are the 3D vector component values.
X,Y,Z is the original non-compensated tool position

U = D * Vx + X
V = D * Vy + Y
W = D * Vz + Z

[0023] The use of the L code represents a conical angle measured from the tool tip point to the nearest obstacle from a flat 2D plane. If the user specifies an angle after LLIMIT, then the tool position move may be completely omitted by the machine if an obstacle is encountered on the part surface in order to automatically avoid gouging as part of the central set of math routine algorithms.

Element Title: Gouge Subroutine

L!= the value given after the LLIMIT command.

L = (D / Sin(L!))

If L < 0 Then skip this move.

Else, combine this value with the D distance value to arrive at a new distance to compensate.

D=D+L

[0029] As such this set of central math routine algorithms using variables to show the math matrix calculation is shown below:

Element Title: Central Subroutine

```
Cz = Cos(Rz): Sz = Sin(Rz): Cx = Cos(Rx): Sx = Sin(Rx): Cy = Cos(Ry): Sy = Sin(Ry)

 'Z rotate, counter clockwise
 X1 = U * Cz + V * Sz: Y1 = U * -Sz + V * Cz: Z1 = W
 'Y rotate, back
 X2 = X1: Y2 = Y1 * Cx + Z1 * -Sx: Z2 = Y1 * Sx + Z1 * Cx
 'X rotate, left
 U = X2 * Cy + Z2 * -Sy: V = Y2: W = X2 * Sy + Z2 * Cy
```

[0030] As referred to in Claims 9 and 9a, the database is an internal list for storage of events, variables, conditions and positions kept in standard computer random access memory. The format for this information is kept in multiple sequential standard matrix arrays. The data is accessed randomly as needed. The formats are double, matrix array as shown below for all collected and gathered user data, variables and positions:

Element Title: Database Subroutine

```
Position1(X,Y,Z,4,5,6,7,8)
Position2(X,Y,Z,4,5,6,7,8)
Position3(X,Y,Z,4,5,6,7,8)
Etc... to Nth Position
Position Nth(X,Y,Z,4,5,6,7,8)

VariableData1(Var1,Var2,Var3,Var4,Var5,Var6,Var7,Var8)
VariableData2(Var1,Var2,Var3,Var4,Var5,Var6,Var7,Var8)
VariableData3(Var1,Var2,Var3,Var4,Var5,Var6,Var7,Var8)
Etc... to Nth
VariableData Nth(Var1,Var2,Var3,Var4,Var5,Var6,Var7,Var8)
```

UserData1(User1,User2,User3,User4,User5,User6,User7,User8)
UserData2(User1,User2,User3,User4,User5,User6,User7,User8)
UserData3(User1,User2,User3,User4,User5,User6,User7,User8)
Etc… to Nth
UserData Nth(User1,User2,User3,User4,User5,User6,User7,User8)

[0031] Presents a group of elements titled as the collection of mathematical subroutine elements and enumerated here as Paragraphs [0031] through [0054]. The provided flowchart in block diagram form, FIG 10, recites all of the elements, components and steps completely constituting every aspect of the technology elements enumerated as Paragraphs [0030] titled as Intelligent Database subroutine and Database subroutine which calls, ties to and works together with the group of elements titled the collection of mathematical subroutine elements enumerated as Paragraphs [0031] through [0054] and specifically linked to and shown in FIG 10 of the block diagram as it interacts with the Element titled DbAtr enumerated as paragraph [0043] , Element titled DbGet enumerated as paragraph [0044], Element titled DbSet enumerated as paragraph [0045] and Element titled DbSetAtrCur enumerated as paragraph [0046].

Subroutine Element Form_Load
Reads in all data from user input boxes from FIG 1 and stores them into the Database Element as described and enumerated as paragraph [0030].

```
Private Sub Form_Load()

On Local Error GoTo LloadErr

IniDir$ = Environ$("AS3000"): If Right$(IniDir$, 1) <> "\" Then IniDir$ = IniDir$ + "\"
Call GloRead

Call PrevInst

If Command$ <> "LAUNCH FROM CNC ONLY" Then MsgBox "You must launch this
from the CNC": End

 ShowDone% = 0
 'Call IniRead("CNCTOOL.INI", "FORM")
 'Call IniDat("TOP", T$): CNCtool.Top = Val(T$)
 'Call IniDat("LEFT", T$): CNCtool.Left = Val(T$)
```

```
 'Call IniDat("HEIGHT", T$): CNCtool.Height = Val(T$)
 'Call IniDat("WIDTH", T$): CNCtool.Width = Val(T$)
 ShowDone% = 1

If Tune% = 1 Then Sounds.MMControl1.Enabled = True

SSPanel6.Top = 60: SSPanel6.Left = 6540


If Mach$ = "LATHE" Then
  ' OLD For Standard Lathe
  'SSPanel1.Caption = "                    Tool Parameters
Tool Nose  Z axis    X axis    Custom    Wear   Custom1 Custom2        Radius     Horz
Vert    3rd axis"
  'SSPanel2.Caption = "                    Machine Offsets
Z       X       3       4       5       6"
  'SSPanel6.Caption = "Tool Definitions (Solid Mode Only)  Corner  Bottom   Side
Length Type radius    angle   angle                     "
  'Label5.Caption = "Z": Label6.Caption = "X": Label7.Caption = "3"
  ' OLD For Vertical Turning Lathe
  'SSPanel1.Caption = ""
  'SSPanel2.Caption = "                    Machine Offsets
X               Z       4       5       6"
  'SSPanel6.Caption = "Tool Definitions (Solid Mode Only)  Corner  Bottom   Side
Length Type radius    angle   angle                     "
  'Label5.Caption = "X": Label6.Caption = " ": Label7.Caption = "Z"
  'New LATHE
  Label5.Caption = "Z": Label6.Caption = "X": Label7.Caption = "3"
  SSPanel1.Caption = "                    Tool Parameters
Tool Nose  Z axis   X axis  3rd axis  Wear  Custom1Custom2       Radius     Horz
Vert"
  SSPanel2.Caption = "                    Machine Offsets
Z       X       3       4       5       6       7       8"
  SSPanel6.Caption = " Tool Definitions  (Solid Mode Only)  Corner  Bottom   Side
Length   Type   radius  angle    angle             "
 Else
  'OLD Standard mill
  'SSPanel1.Caption = "                    Tool Parameters
Size     Horz      Vert     Height    Wear  Custom1 Custom2"
  'SSPanel2.Caption = "                    Machine Offsets
X       Y       Z       4       5       6"
  'SSPanel6.Caption = "Tool Definitions  (Solid Mode Only) Corner  Bottom   Side
Length Type  radius   angle   angle                     "
  'New Mill
  Label5.Caption = "X": Label6.Caption = "Y": Label7.Caption = "Z"
```

```
 SSPanel1.Caption = "                              Tool Parameters
Size     Horz     Vert    Height   Wear  Custom1Custom2"
 SSPanel2.Caption = "                                          Machine Offsets
X         Y         Z         4         5         6         7         8"
 SSPanel6.Caption = " Tool Definitions  (Solid Mode Only)  Corner  Bottom   Side
Length  Type  radius  angle    angle                    "
End If

ToolPage% = 0
ToolDef% = 0
ToolDescrip% = 0
ToolPics% = 0

If Dir$(IniDir$ + "CNC\TOOLDEF.FIL") <> "" Then ToolDef% = 1
If Dir$(IniDir$ + "CNC\TOOLCUS.FIL") <> "" Then ToolCus% = 1
If Dir$(IniDir$ + "CNC\TOOLDESP.FIL") <> "" Then ToolDescrip% = 1
If Dir$(IniDir$ + "CNC\TOOLPICS.FIL") <> "" Then ToolPics% = 1

F1% = FreeFile: Open IniDir$ + "CNC\TOOL.FIL" For Input As #F1%
If ToolDef% = 1 Then F2% = FreeFile: Open IniDir$ + "CNC\TOOLDEF.FIL" For Input
As #F2%
If ToolCus% = 1 Then F3% = FreeFile: Open IniDir$ + "CNC\TOOLCUS.FIL" For
Input As #F3%
If ToolDescrip% = 1 Then F4% = FreeFile: Open IniDir$ + "CNC\TOOLDESP.FIL" For
Input As #F4%
If ToolPics% = 1 Then F5% = FreeFile: Open IniDir$ + "CNC\TOOLPICS.FIL" For
Input As #F5%

 For Cnt% = 0 To 9
  Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text1(Cnt%).Text =
Trim$(Dum$) 'Size
  Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text2(Cnt%).Text =
Trim$(Dum$) 'Horz
  Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text3(Cnt%).Text =
Trim$(Dum$) 'Vert
  Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text4(Cnt%).Text =
Trim$(Dum$) 'Height
  Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text5(Cnt%).Text =
Trim$(Dum$) 'Wear
  If ToolDef% = 1 Then
   Input #F2%, Dum$: Dum$ = Format$(Dum$, "####0.0#####"): Text16(Cnt%).Text =
Trim$(Dum$) 'Corner Radius
   Input #F2%, Dum$: Dum$ = Format$(Dum$, "####0.0#####"): Text17(Cnt%).Text =
Trim$(Dum$) 'Bottom Angle
   Input #F2%, Dum$: Dum$ = Format$(Dum$, "####0.0#####"): Text18(Cnt%).Text =
Trim$(Dum$) 'Side Angle
```

```
  Input #F2%, Dum$: Dum$ = Format$(Dum$, "####0.0#####"): Text19(Cnt%).Text =
Trim$(Dum$) 'Length
  Input #F2%, Dum$: Text20(Cnt%).Text = Trim$(Dum$) 'ToolType
 End If
 If ToolCus% = 1 Then
  Input #F3%, Dum$: Text23(Cnt%).Text = Trim$(Dum$) 'Custom1
  Input #F3%, Dum$: Text24(Cnt%).Text = Trim$(Dum$) 'Custom2
 End If
 If ToolDescrip% = 1 Then
  Input #F4%, Dum$: Text26(Cnt%).Text = Trim$(Dum$) 'Desp
  Input #F4%, Dum$: Text25(Cnt%).Text = Trim$(Dum$) 'Time
 End If
 'ToolPics% is F5% not needed here
Next Cnt%

 Seek F1%, 1
If ToolDef% = 1 Then Seek F2%, 1
If ToolCus% = 1 Then Seek F3%, 1
If ToolDescrip% = 1 Then Seek F4%, 1
If ToolPics% = 1 Then Seek F5%, 1

'Load rest of tool info into arrays ' MaxTools%
For Cnt% = 1 To MaxTools%
 Input #F1%, ToolSize!(Cnt%)
 Input #F1%, ToolHorz!(Cnt%)
 Input #F1%, ToolVert!(Cnt%)
 Input #F1%, ToolHeight!(Cnt%)
 Input #F1%, ToolWear!(Cnt%)
 If ToolDef% = 1 Then
   Input #F2%, ToolCorRad!(Cnt%)
   Input #F2%, ToolBotAng!(Cnt%)
   Input #F2%, ToolSideAng!(Cnt%)
   Input #F2%, ToolLength!(Cnt%)
   Input #F2%, ToolType!(Cnt%)
 End If
 If ToolCus% = 1 Then
   Input #F3%, ToolCustom1!(Cnt%)
   Input #F3%, ToolCustom2!(Cnt%)
 End If
 If ToolDescrip% = 1 Then
   Input #F4%, ToolDesp$(Cnt%)
   Input #F4%, ToolTime!(Cnt%)
 End If
 If ToolPics% = 1 Then
   Input #F5%, Dum$: ToolPhoto$(Cnt%) = Trim$(Dum$) 'Desp
 End If
```

```
Next Cnt%

Close F1%, F2%, F3%, F4%, F5%

F1% = FreeFile: Open IniDir$ + "CNC\TOOLOPT.FIL" For Input As #F1%
' machine offsets
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text6(0).Text =
Trim$(Dum$) 'X
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text6(1).Text =
Trim$(Dum$) 'Y
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text6(2).Text =
Trim$(Dum$) 'Z
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text6(3).Text =
Trim$(Dum$) '4
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text6(4).Text =
Trim$(Dum$) '5
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text6(5).Text =
Trim$(Dum$) '6
'Input #F1%, Dum$: Text6(6).Text = Trim$(Dum$) '7 see end of file line 23
'Input #F1%, Dum$: Text6(7).Text = Trim$(Dum$) '8

'Fixture offsets
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text7(0).Text =
Trim$(Dum$) 'X G54
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text7(1).Text =
Trim$(Dum$) 'Y
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text7(2).Text =
Trim$(Dum$) 'Z
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text7(3).Text =
Trim$(Dum$) '4
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text7(4).Text =
Trim$(Dum$) '5
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text7(5).Text =
Trim$(Dum$) '6
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text8(0).Text =
Trim$(Dum$) 'X G55
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text8(1).Text =
Trim$(Dum$) 'Y
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text8(2).Text =
Trim$(Dum$) 'Z
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text8(3).Text =
Trim$(Dum$) '4
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text8(4).Text =
Trim$(Dum$) '5
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text8(5).Text =
Trim$(Dum$) '6
```

```
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text9(0).Text = Trim$(Dum$) 'X G56
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text9(1).Text = Trim$(Dum$) 'Y
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text9(2).Text = Trim$(Dum$) 'Z
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text9(3).Text = Trim$(Dum$) '4
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text9(4).Text = Trim$(Dum$) '5
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text9(5).Text = Trim$(Dum$) '6
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text10(0).Text = Trim$(Dum$) 'X G57
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text10(1).Text = Trim$(Dum$) 'Y
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text10(2).Text = Trim$(Dum$) 'Z
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text10(3).Text = Trim$(Dum$) '4
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text10(4).Text = Trim$(Dum$) '5
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text10(5).Text = Trim$(Dum$) '6
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text11(0).Text = Trim$(Dum$) 'X G58
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text11(1).Text = Trim$(Dum$) 'Y
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text11(2).Text = Trim$(Dum$) 'Z
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text11(3).Text = Trim$(Dum$) '4
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text11(4).Text = Trim$(Dum$) '5
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text11(5).Text = Trim$(Dum$) '6
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text12(0).Text = Trim$(Dum$) 'X G59
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text12(1).Text = Trim$(Dum$) 'Y
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text12(2).Text = Trim$(Dum$) 'Z
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text12(3).Text = Trim$(Dum$) '4
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text12(4).Text = Trim$(Dum$) '5
```

```
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text12(5).Text =
Trim$(Dum$) '6

Input #F1%, Dum$: SSCheck1.Value = Val(Dum$) 'Dry Run
Input #F1%, Dum$: SSCheck2.Value = Val(Dum$) 'BitMap G code
Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text13.Text =
Trim$(Dum$) 'Tolerance
Input #F1%, Dum$: Text14.Text = Trim$(Dum$) 'Block Skip Char
Input #F1%, Dum$: Text15.Text = Trim$(Dum$) 'Teach Filename

Input #F1%, Dum$: SSOption1(0).Value = Val(Dum$) 'Absolute
Input #F1%, Dum$: SSOption1(1).Value = Val(Dum$) 'Incremental
Input #F1%, Dum$: SSOption1(2).Value = Val(Dum$) 'R code

23 ' extra tool options
 Text21.Text = "0"
  Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text21.Text =
Trim$(Dum$) ' Solid stock Z begin
 Text22.Text = "1"
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text22.Text =
Trim$(Dum$) ' Extra Stock
 SSCheck3.Value = 0
 Input #F1%, Dum$: SSCheck3.Value = Val(Dum$) 'Graphics: Solids vs Wire Frame
 SSCheck4.Value = 0
 Input #F1%, Dum$: SSCheck4.Value = Val(Dum$) ' WireTrace
 If SSCheck3.Value = True Then
   SSCheck4.Value = False: SSCheck4.Visible = False
  Else
   SSCheck4.Visible = True
 End If

 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text6(6).Text =
Trim$(Dum$) '7
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text6(7).Text =
Trim$(Dum$) '8
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text7(6).Text =
Trim$(Dum$) '7 G54
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text7(7).Text =
Trim$(Dum$) '8
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text8(6).Text =
Trim$(Dum$) '7 G55
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text8(7).Text =
Trim$(Dum$) '8
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text9(6).Text =
Trim$(Dum$) '7
```

```
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text9(7).Text =
Trim$(Dum$) '7
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text10(6).Text =
Trim$(Dum$) '7
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text10(7).Text =
Trim$(Dum$) '8
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text11(6).Text =
Trim$(Dum$) '7
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text11(7).Text =
Trim$(Dum$) '8
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text12(6).Text =
Trim$(Dum$) '7 G59
 Input #F1%, TemN!: Dum$ = Format$(TemN!, "####0.0#####"): Text12(7).Text =
Trim$(Dum$) '8

 Close F1%

Help$ = "CNCTOOL.Hlp"
Exit Sub

Lload:
Close F1%
Exit Sub

LloadErr:
If Erl = 23 Then Resume Lload
MsgBox Str$(Err), 48, "Error"
End

End Sub
```

[0032] Subroutine Element GloRead

Reads in all global and public data from user input boxes plus any proprietary settings from FIG 1 and stores them into the Database Element as described in and enumerated as paragraph [0030].

```
Sub GloRead()

On Local Error GoTo GloReadERR:

  F% = FreeFile: G% = 0
  Open IniDir$ + "ini\PLANES.FIL" For Input As #F%
```

```
  Do
    G% = G% + 1
    Input #F%, PlnBack!(G%)
    Input #F%, PlnLeft!(G%)
    Input #F%, PlnCw!(G%)
  Loop Until G% = 256

Close F%

  F% = FreeFile
  Open IniDir$ + "ini\GLOBAL.FIL" For Input As #F%

  Line Input #F%, BitMap$
  Line Input #F%, Sound$
  Line Input #F%, Ram$
   K% = InStr(Ram$, "\"): If K% = 0 Then Ram$ = Ram$ + "\"
  Line Input #F%, FileW$
   K% = InStr(FileW$, "\"): If K% = 0 Then FileW$ = FileW$ + "\"
  Line Input #F%, Filet$
   K% = InStr(Filet$, "\"): If K% = 0 Then Filet$ = Filet$ + "\"
  Line Input #F%, Pass1$
  Line Input #F%, Pass2$
  Line Input #F%, Pass3$
  Line Input #F%, CurFile$
  Line Input #F%, Help$
  Input #F%, Max%
  Input #F%, Layer%
  Input #F%, Path%
  Input #F%, BAD%
  Input #F%, Plane%
  Line Input #F%, Mach$
  Line Input #F%, Ver$
  Input #F%, Scan%
  Input #F%, Colr%
  Input #F%, Tune%
  Input #F%, Speed!
  Input #F%, Feed!
  Input #F%, Tool!
  Input #F%, Dia!
  Input #F%, Rapid!
  Input #F%, mode%
  Input #F%, Redraw%
  Input #F%, Metric%
  Input #F%, T2D%
  Input #F%, Toler!
  Input #F%, SHIFTA!
```

```
  Input #F%, FirstHelp%
  Input #F%, HiLitePath%

  Close #F%
  TolerSurf! = Toler! * 20
  'Level% = Level% / 33

  Exit Sub

GloReadERR:
Close F%
If Err = 53 Then MsgBox IniDir$ + "ini\Global.Fil Not Found", 65536 + 16, "Error": End
MsgBox "Can't Open " + IniDir$ + "ini\Global.Fil", 65536 + 16, "Error " + Str$(Err) +
":" + Str$(Erl): End
End

End Sub
```

[0033] Subroutine Element ANG2VEC

Returns angle between two vectors and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub ANG2VEC(SubVx1!, SubVy1!, SubVz1!, SubVx2!, SubVy2!, SubVz2!, SubAng!)
Vx1! = SubVx1!: Vy1! = SubVy1!: Vz1! = SubVz1!: Vx2! = SubVx2!: Vy2! = SubVy2!:
Vz2! = SubVz2!
Call RCOS(T!)
'If T! < Toler! Then T! = 360 ' leave to calling sub
SubAng! = Abs(T!)
End Sub
```

[0034] Subroutine Element AngInArc

Tells if Angle given falls between arc angles and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub AngInArc(SubSTang!, SubEndAng!, SubTestAng!, SUBRad!, SUBHIT%)
```

```
SA! = SubSTang!: EA! = SubEndAng!: TA! = SubTestAng!: R! = SUBRad!: HIT% = 0
Call TolAng(R!, TOL!)
If TA! = 360 Then TA! = 0
If EA! = 360 Then EA! = 0
If SA! = 0 Then SA! = 360
If TA! = 0 And SA! = 360 And EA! <= SA! Then TA! = 360
If EA! <= SA! And TA! + TOL! >= EA! And TA! - TOL! <= SA! Then
 HIT% = 1
 If TA! - TOL! <= SA! And TA! + TOL! >= EA! Then HIT% = 2
End If
If EA! >= SA! Then
  If TA! + TOL! >= EA! Or TA! - TOL! <= SA! Then
   HIT% = 1
   If TA! - TOL! <= EA! Or TA! + TOL! >= SA! Then HIT% = 2
  End If
End If
SUBHIT% = HIT%
End Sub
```

[0035] Subroutine Element AngVec

Changes XYZ vectors to real Angles relative to plane and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub AngVec(SubVx!, SubVy!, SubVz!, SubPLn%)
Vx! = SubVx!: Vy! = SubVy!: Vz! = SubVz!
Call Rsin(Vx!): Call Rsin(Vy!): Call Rsin(Vz!)
B! = PlnBack!(SubPLn%): L! = PlnLeft!(SubPLn%): C! = PlnCw!(SubPLn%)
Call View2Vec(B!, L!, C!, Pvx!, Pvy!, Pvz!)
SubVx! = Vx! - (90 - Pvx!)
SubVy! = Vy! - (90 - Pvy!)
SubVz! = Vz! - (90 - Pvz!)
End Sub
```

[0036] Subroutine Element Arc3pt3D

Finds center of arc and radius given 3 points and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub Arc3pt3D(SubX1!, SubY1!, SubZ1!, SubX2!, SubY2!, SubZ2!, SUBX3!, SUBY3!,
SUBZ3!, SUBI!, SubJ!, SubK!, SubR!, SubEr%)

SubEr% = 0

On Local Error GoTo LArc3pt3D
X1! = SubX1!: Y1! = SubY1!: Z1! = SubZ1!: X2! = SubX2!: Y2! = SubY2!: Z2! =
SubZ2!: X3! = SUBX3!: Y3! = SUBY3!: Z3! = SUBZ3!
A! = (Y2! - Y1!) * (Z3! - Z2!) - (Y3! - Y2!) * (Z2! - Z1!)
B! = (X3! - X2!) * (Z2! - Z1!) - (X2! - X1!) * (Z3! - Z2!)
C! = (X2! - X1!) * (Y3! - Y2!) - (X3! - X2!) * (Y2! - Y1!)
B1! = 1: C1! = (-1 / C!) * ((A1! * A!) + (B1! * B!))
TemN! = ((X3! - X1!) * C! - (Z3! - Z1!) * A!): If TemN! = 0 Then TemN! = 0.00001
A2! = ((Z3! - Z1!) * B! - (Y3! - Y1!) * C!) / TemN!: B2! = 1
S! = ((X3! - X2!) * B1! - (Y3! - Y2!) * A1!) / (((A1! * B2!) - (A2! * B1!)) * 2)
X! = ((X3! + X1!) / 2) + (A2! * S!)
Y! = ((Y3! + Y1!) / 2) + (B2! * S!)
Z! = ((Z3! + Z1!) / 2) + (C2! * S!)
XX! = (X! - X1!): YY! = (Y! - Y1!): ZZ! = (Z! - Z1!)
R! = Sqr(XX! * XX! + YY! * YY! + ZZ! * ZZ!)
If Metric% = 0 And R! > 1000 Then SubEr% = 1
If Metric% = 1 And R! > 25400 Then SubEr% = 1
SUBI! = X!: SubJ! = Y!: SubK! = Z!: SubR! = R!
Exit Sub

LArc3pt3D:
'Call Play("ERROR"): MsgBox "The 3 Positions are a Straight Line", 65536 +48, "Can
Not Make ARC"
 SubEr% = 1
Resume Larc3pt3D5

Larc3pt3D5:
End Sub
```

[0037] Subroutine Element ArcEnd

Calculates the ends of arc positions in 3D and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub ArcEnd(SUBI!, SubJ!, SubK!, SubR!, SubS!, SubE!, SubPLn%, SubX1!, SubY1!,
SubZ1!, SubX2!, SubY2!, SubZ2!)
  i! = SUBI!: J! = SubJ!: K! = SubK!: R! = SubR!: S! = SubS!: E! = SubE!: P% =
SubPLn%
  S! = S! * Radian!: E! = E! * Radian!
  X1! = R! * Sin(S!): Y1! = R! * Cos(S!): Z1! = 0
  X2! = R! * Sin(E!): Y2! = R! * Cos(E!): Z2! = 0
  If P% > 0 Then Call R2P(X1!, Y1!, Z1!, P%): Call R2P(X2!, Y2!, Z2!, P%)
  SubX1! = X1! + i!: SubY1! = Y1! + J!: SubZ1! = Z1! + K!: SubX2! = X2! + i!:
SubY2! = Y2! + J!: SubZ2! = Z2! + K!
End Sub
```

[0038] Subroutine Element ArcLen

Calculates the length of arc positions in 3D and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub ArcLen(SubS!, SubE!, SubR!, SubL!)
  S! = SubS!: E! = SubE!: R! = SubR!
  i! = S! - E!: If i! <= 0 Then i! = i! + 360
  SubL! = (R! * i! * 3.1415926) / 180
End Sub
```

[0039] Subroutine Element BiSectAng

Calculate Bisected 3D angles and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub BiSectAng(SUBSA!, SUBEA!, SubNew!)
 Sang! = SUBSA!: Eang! = SUBEA!
 If Sang! < Eang! Then Sang! = Sang! + 360
 N! = (Sang! - Eang!) / 2
 N! = N! + Eang!
 If N! > 360 Then N! = N! - 360
 SubNew! = N!
End Sub
```

[0040] Subroutine Element BISECVEC

Calculate Bisected 3D vectors and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub BISECVEC(SubX1!, SubY1!, SubZ1!, SubX2!, SubY2!, SubZ2!, SubVx!, SubVy!,
SubVz!)
X1! = SubX1!: Y1! = SubY1!: Z1! = SubZ1!: X2! = SubX2!: Y2! = SubY2!: Z2! =
SubZ2!
 Q! = Sqr(A! * A! + B! * B! + C! * C!)
If Abs(Q!) < 0.0002 Then SubVx! = 0: SubVy! = 0: SubVz! = 0: Exit Sub
SubVx! = A! / Q!: SubVy! = B! / Q!: SubVz! = C! / Q!
End Sub
```

[0041] Subroutine Element CrLnIfInt

Calculates 3D Circle/Line Intersections and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub CrLnIfInt(SUBI!, SubJ!, SUBRad!, SubSang!, SubEang!, SubX1!, SubY1!, SubX2!,
SubY2!, SubX1st!, SubY1st!, SubHit1%, SubX2nd!, SubY2nd!, SubHit2%)
i! = SUBI!: J! = SubJ!: R! = SUBRad!: S! = SubSang!: E! = SubEang!
X1! = SubX1!: Y1! = SubY1!: X2! = SubX2!: Y2! = SubY2!: SubHit1% = 0: SubHit2%
= 0

Call MATH(X1!, Y1!, i!, J!, 1, A!, R!, Em$, Er%, XA!, YA!, XB!, YB!): If Er% = 1
Then Exit Sub

'test 1st intersection
SubX1st! = XA!: SubY1st! = YA!
Call PtInCr(i!, J!, R!, S!, E!, XA!, YA!, Hit1%)
If Hit1% = 1 And Hit2% > 0 Then SubHit1% = 1
If Hit1% = 2 And Hit2% > 0 Then SubHit1% = 2

'test 2nd intersection
SubX2nd! = XB!: SubY2nd! = YB!
Call PTINLN(X1!, Y1!, X2!, Y2!, XB!, YB!, Hit2%)
If Hit1% = 1 And Hit2% > 0 Then SubHit2% = 1
If Hit1% = 2 And Hit2% > 0 Then SubHit2% = 2
End Sub
```

[0042] Subroutine Element CrossErr

Calculates errors in tool comp and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub CrossErr(SubPLn%, SubVx!, SubVy!, SubVz!)
   P% = SubPLn%
   B! = PlnBack!(P%): L! = PlnLeft!(P%): C! = PlnCw!(P%)
   Call View2Vec(B!, L!, C!, Vx!, Vy!, Vz!)
```

Appl. No. 10/079,309

```
  SubVx! = Vx!: SubVy! = Vy!: SubVz! = Vz!
End Sub
```

[0043] Subroutine Element DbAtr

Database element to store geometry properties, error, conditions and positions. Works with the Element titled Intelligent Database subroutine enumerated as paragraph [0030].

```
Sub DbAtr(QATR1%, QATR2%, QATR3%, QATR4%, QATR5%, QATR6%,
QATR7%, QATR8%, QATR9%)

  ' If ItemNumber% <=0 then Close file

  'QATR1% = Item number
  'QATR2% = Function
  'QATR3% = Hot property
  'QATR4% = Entity type
  'QATR5% = Path number
  'QATR6% = Layer numbe
  'QATR7% = Line style
  'QATR8% = Position
  'QATR9% = Error

  Qitem% = QATR1%
  If Qitem% <= 0 Then Close 153: DbOpen% = 0: Exit Sub
  If DbOpen% = 0 Then DbOpen% = 1: Close 153: Open Filet$ + "DATABASE.FIL"
For Random As #153 Len = Len(RecFile)

  Select Case QATR2%
    Case 0   'Set parameters
        TemL& = Len(RecFile)
        If (Qitem% * TemL&) <= LOF(153) Then Get #153, Qitem%, RecFile 'Must Get
other things in RecFile before write
        RecFile.aaHot = QATR3%
        RecFile.aaType = QATR4%
        RecFile.aaPath = QATR5%
        RecFile.aaLayer = QATR6%
        RecFile.aaStyle = QATR7%
        RecFile.aaColor = QATR8%
        RecFile.aaPlane = QATR9%
       Put #153, Qitem%, RecFile

    Case 1   'Get parameters
```

```
    Get #153, Qitem%, RecFile
       QATR3% = RecFile.aaHot
       QATR4% = RecFile.aaType
       QATR5% = RecFile.aaPath
       QATR6% = RecFile.aaLayer
       QATR7% = RecFile.aaStyle
       QATR8% = RecFile.aaColor
       QATR9% = RecFile.aaPlane
  End Select

End Sub
```

[0044] Subroutine Element DbGet

Gets Database item coordinate, property and position from random file. Works with the Element titled Intelligent Database subroutine enumerated as paragraph [0030].

```
Sub DbGet(SubItem%, SubX1!, SubY1!, SubZ1!, SubX2!, SubY2!, SubZ2!)

' If ItemNumber% <=0 then Close file


On Local Error GoTo dbgetERR:
TT% = SubItem%

If TT% <= 0 Then Close #153: DbOpen% = 0: Exit Sub

If DbOpen% = 0 Then DbOpen% = 1: Close 153: Open Filet$ + "DATABASE.FIL" For
Random As #153 Len = Len(RecFile)

Get #153, TT%, RecFile
 SubX1! = RecFile.aaX1
 SubY1! = RecFile.aaY1
 SubZ1! = RecFile.aaZ1
 SubX2! = RecFile.aaX2
 SubY2! = RecFile.aaY2
 SubZ2! = RecFile.aaZ2

Exit Sub

dbgetERR:
Close 153: DbOpen% = 0
```

```
If Err = 53 Then MsgBox Filet$ + "DATABASE.FIL Not Found", 65536 + 16, "Error":
End
MsgBox "Can't Open " + Filet$ + "DATABASE.FIL", 65536 + 16, "Error " + Str$(Err) +
":" + Str$(Erl): End
End

End Sub
```

[0045] Subroutine Element DbSet

Sets Database item coordinate, property and position from random file. Works with the Element titled Intelligent Database subroutine enumerated as paragraph [0030].

```
Sub DbSet(SubItem%, SubX1!, SubY1!, SubZ1!, SubX2!, SubY2!, SubZ2!)

' Sets Database item coord into random file
' If ItemNumber% <=0 then Close file

On Local Error GoTo DbSetERR:
TT% = SubItem%
If TT% <= 0 Then Close #153: DbOpen% = 0: Exit Sub

If DbOpen% = 0 Then DbOpen% = 1: Close #153: Open Filet$ + "DATABASE.FIL" For
Random As #153 Len = Len(RecFile)

 TemL& = Len(RecFile)
 If (TT% * TemL&) <= LOF(153) Then Get #153, TT%, RecFile  'Must Get other things
in RecFile
 RecFile.aaX1 = SubX1!
 RecFile.aaY1 = SubY1!
 RecFile.aaZ1 = SubZ1!
 RecFile.aaX2 = SubX2!
 RecFile.aaY2 = SubY2!
 RecFile.aaZ2 = SubZ2!
 Put #153, TT%, RecFile

 Exit Sub

 DbSetERR:
 Close 153: DbOpen% = 0
 If Err = 53 Then MsgBox Filet$ + "DATABASE.FIL Not Found", 65536 + 16, "Error":
 End
 MsgBox "Can't Open " + Filet$ + "DATABASE.FIL", 65536 + 16, "Error " + Str$(Err) +
 ":" + Str$(Erl): End
```

End

End Sub

[0046] Subroutine Element DbSetAtrCur

Stores, retrieves and records current database variables in memory to work together with the Element titled Intelligent Database subroutine enumerated as paragraph [0030].

```
Sub DbSetAtrCur(SubType%)

TT% = Max%
T% = SubType%
H% = 1
Pth% = 0
L% = Layer%
If SubType% = 2 Then S% = Style% Else S% = 0
C% = Colr%
Pln% = Plane%
Call DbAtr(TT%, 0, H%, T%, Pth%, L%, S%, C%, Pln%)

End Sub
```

[0047] Subroutine Element DefPln3pts

Finds 3D plane vector normals and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub DefPln3pts(SubPx1!, SubPy1!, SubPz1!, SubPx2!, SubPy2!, SubPz2!, SubPx3!,
SubPy3!, SubPz3!, SubVx!, SubVy!, SubVz!)
Px1! = SubPx1!: Py1! = SubPy1!: Pz1! = SubPz1!: Px2! = SubPx2!: Py2! = SubPy2!:
Pz2! = SubPz2!: Px3! = SubPx3!: Py3! = SubPy3!: Pz3! = SubPz3!
Call Vector(Px2!, Py2!, Pz2!, Px3!, Py3!, Pz3!, Vx1!, Vy1!, Vz1!, Vd1!)
Call Vector(Px2!, Py2!, Pz2!, Px1!, Py1!, Pz1!, Vx2!, Vy2!, Vz2!, Vd2!)
Call CROSSVEC(Vx1!, Vy1!, Vz1!, Vx2!, Vy2!, Vz2!, Vx!, Vy!, Vz!)
SubVx! = Vx!: SubVy! = Vy!: SubVz! = Vz!
End Sub
```

[0048] Subroutine Element LnTan2Arc

Calculates 3D Line Tangent to Arc at Angle Intersections and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub LnTan2Arc(SUBI!, SubJ!, SubR!, SubAng!, SubCx!, SubCy!, SubIntX!, SubIntY!)
i! = SUBI!: J! = SubJ!: R! = SubR!: A! = SubAng!: CX! = SubCx!: CY! = SubCy!
X1! = i!: Y1! = J!: A1! = A! - 90: Call Fixang(A1!): Call Polar(X1!, Y1!, A1!, R!)
X2! = i!: Y2! = J!: A2! = A! + 90: Call Fixang(A2!): Call Polar(X2!, Y2!, A2!, R!)
SubIntX! = X1!: SubIntY! = Y1!
Call Dis(CX!, CY!, X1!, Y1!, D1!)
Call Dis(CX!, CY!, X2!, Y2!, D2!)
If D2! < D1! Then SubIntX! = X2!: SubIntY! = Y2!
End Sub
```

[0049] Subroutine Element LnTan2Arcs

Calculates 3D Line Tangent to two Arcs at Angle Intersections and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub LnTan2Arcs(SUBI1!, SubJ1!, SubR1!, SUBI2!, SubJ2!, SubR2!, SubCx1!, SubCy1!,
SubCx2!, SubCy2!, SubIntX1!, SubIntY1!, SubIntX2!, SubIntY2!, SubEr%)
I1! = SUBI1!: J1! = SubJ1!: R1! = SubR1!
I2! = SUBI2!: J2! = SubJ2!: R2! = SubR2!
Cx1! = SubCx1!: Cy1! = SubCy1!: SubEr% = 0
Cx2! = SubCx2!: Cy2! = SubCy2!
Call Dis(I1!, J1!, I2!, J2!, D!)
'If D! < R1! Or D! < R2! Then SubEr% = 1: Exit Sub
Call Ang(I1!, J1!, I2!, J2!, A!)
Call Vector(I1!, J1!, 0, i!, J!, 0, Vx!, Vy!, Vz!, Vd!) ' Vector perperndicular to angle
between arc centers
Call DISVEC(I1!, J1!, 0, Cx2!, Cy2!, 0, Vx!, Vy!, Vz!, D2!)
TheSame% = 0: F! = (R1! + R2!) / D! ' If both crosshairs are on different sides
If D1! < 0 And D2! < 0 Then TheSame% = 1: F! = (R1! - R2!) / D! ' If both crosshairs
are on same side
```

```
If D1! > 0 And D2! > 0 Then TheSame% = 1: F! = (R1! - R2!) / D! ' If both crosshairs
are on same side
Call Rsin(F!): F! = 90 - Abs(F!) ' to get angle from arc center

Select Case TheSame%
Case 0
 'First arc
 T! = A! + F!: X1! = I1!: Y1! = J1!: Call Polar(X1!, Y1!, T!, R1!)
 T! = A! - F!: X2! = I1!: Y2! = J1!: Call Polar(X2!, Y2!, T!, R1!)
 SubIntX1! = X1!: SubIntY1! = Y1!
 Call Dis(Cx1!, Cy1!, X1!, Y1!, D1!)
 Call Dis(Cx1!, Cy1!, X2!, Y2!, D2!): If D2! < D1! Then SubIntX1! = X2!: SubIntY1! =
Y2!
 'Second Arc
 A! = A! + 180: Call Fixang(A!)
 T! = A! + F!: X1! = I2!: Y1! = J2!: Call Polar(X1!, Y1!, T!, R2!)
 T! = A! - F!: X2! = I2!: Y2! = J2!: Call Polar(X2!, Y2!, T!, R2!)
 SubIntX2! = X1!: SubIntY2! = Y1!
 Call Dis(Cx2!, Cy2!, X1!, Y1!, D1!)
 Call Dis(Cx2!, Cy2!, X2!, Y2!, D2!): If D2! < D1! Then SubIntX2! = X2!: SubIntY2! =
Y2!
Case 1
 'First arc
 If R1! < R2! Then A! = A! + 180: Call Fixang(A!)
 T! = A! + F!: X1! = I1!: Y1! = J1!: Call Polar(X1!, Y1!, T!, R1!)
 T! = A! - F!: X2! = I1!: Y2! = J1!: Call Polar(X2!, Y2!, T!, R1!)
 SubIntX1! = X1!: SubIntY1! = Y1!
 Call Dis(Cx1!, Cy1!, X1!, Y1!, D1!)
 'Second Arc
 T! = A! + F!: X1! = I2!: Y1! = J2!: Call Polar(X1!, Y1!, T!, R2!)
 T! = A! - F!: X2! = I2!: Y2! = J2!: Call Polar(X2!, Y2!, T!, R2!)
 SubIntX2! = X1!: SubIntY2! = Y1!
 Call Dis(Cx2!, Cy2!, X1!, Y1!, D1!)
 Call Dis(Cx2!, Cy2!, X2!, Y2!, D2!): If D2! < D1! Then SubIntX2! = X2!: SubIntY2! =
Y2!
End Select

End Sub
```

[0050] Subroutine Element LnTanArcPt

Calculates 3D Line Tangent to arc through point and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub LnTanArcPt(SubPX!, SubPY!, SUBI!, SubJ!, SubR!, SubCx!, SubCy!, SubIntX!,
SubIntY!, SubEr%)
X! = SubPX!: Y! = SubPY!: i! = SUBI!: J! = SubJ!: R! = SubR!: CX! = SubCx!: CY! =
SubCy!: SubEr% = 0
Call Dis(X!, Y!, i!, J!, D!): If D! < R! Then SubEr% = 1: Exit Sub
Call Ang(i!, J!, X!, Y!, A!)
T! = A! + B!: X1! = i!: Y1! = J!: Call Polar(X1!, Y1!, T!, R!)
T! = A! - B!: X2! = i!: Y2! = J!: Call Polar(X2!, Y2!, T!, R!)
SubIntX! = X1!: SubIntY! = Y1!
Call Dis(CX!, CY!, X1!, Y1!, D1!)
Call Dis(CX!, CY!, X2!, Y2!, D2!): If D2! < D1! Then SubIntX! = X2!: SubIntY! = Y2!
End Sub
```

[0051] Subroutine Element MidArc

Finds midway point of 3D arc and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub MidArc(SubPLn%, SubG%, SUBI!, SubJ!, SubK!, SubR!, SubS!, SubE!, SubX!,
SubY!, SubZ!)
P% = SubPLn%: i! = SUBI!: J! = SubJ!: K! = SubK!: R! = SubR!: S! = SubS!: E! =
SubE!
'If SubG% = 2 Then T! = S!: S! = E!: E! = T! ' done by CirConvert
Call BiSectAng(S!, E!, MidAng!)
X! = 0: Y! = 0: Z! = 0
Call Polar(X!, Y!, MidAng!, R!)

If P% > 0 Then Call R2P(X!, Y!, Z!, P%)
X! = i! + X!: Y! = J! + Y!: Z! = K! + Z!

SubX! = X!: SubY! = Y!: SubZ! = Z!
End Sub
```

[0052] Subroutine Element OffCR

Offsets a circle in 3D and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub OffCR(SUBI!, SubJ!, SubK!, SubR!, SubPLn%, SubXp!, SubYp!, SubZp!,
SUBDis!, SubNewRad!, SubEr%)
i! = SUBI!: J! = SubJ!: K! = SubK!: R! = SubR!
P% = SubPLn%: Xp! = SubXp!: Yp! = SubYp!: Zp! = SubZp!: D! = SUBDis!: SubEr% = 0
If P% > 0 Then Call ProjPln(P%, i!, J!, K!, Xp!, Yp!, Zp!) ' adjust chross/hair Z to plane
of arc
X! = Xp! - i!: Y! = Yp! - J!: Z! = Zp! - K!
If T! >= R! Then SubNewRad! = R! + D!
If T! < R! Then SubNewRad! = R! - D!: If SubNewRad! <= 0 Then SubEr% = 1 '
negative but continue
End Sub
```

[0053] Subroutine Element OffLN

Offsets a line in 3D and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub OffLN(SubX1!, SubY1!, SubZ1!, SubX2!, SubY2!, SubZ2!, SubPX!, SubPY!,
SubPZ!, SUBDis!, SubPast%)
X1! = SubX1!: Y1! = SubY1!: Z1! = SubZ1!: X2! = SubX2!: Y2! = SubY2!: Z2! =
SubZ2!
Px! = SubPX!: Py! = SubPY!: Pz! = SubPZ!: D! = SUBDis!

Call PtOnLn(X1!, Y1!, Z1!, X2!, Y2!, Z2!, Px!, Py!, Pz!, IntX!, IntY!, IntZ!, EndX!,
EndY!, EndZ!, Past%)
If Past% = 1 Then Exit Sub

Call Vector(IntX!, IntY!, IntZ!, Px!, Py!, Pz!, Vx!, Vy!, Vz!, Vd!)
Call VecPol3D(X1!, Y1!, Z1!, Vx!, Vy!, Vz!, D!)
SubX1! = X1!: SubY1! = Y1!: SubZ1! = Z1!: SubX2! = X2!: SubY2! = Y2!: SubZ2! = Z2!

End Sub
```

[0054] Subroutine Element Tilt3D

Tilts and rotates a tool for tool comp and works together with and calls the functions in the element titled Database subroutine, Intelligent Database subroutine enumerated as paragraph [0030] and the element titled Central subroutine enumerated as paragraph [0029].

```
Sub Tilt3D(SUB42B!, SUB42L!, SUB42C!, SUB42X!, SUB42Y!, Sub42Z!)
  Rx! = SUB42B!: Ry! = SUB42L!: Rz! = SUB42C!: X! = SUB42X!: Y! = SUB42Y!: Z!
= Sub42Z!
  If Rx! = 0 And Ry! = 0 And Rz! = 0 Then Exit Sub
   ' rotate 3d tool our convention
   Rx! = Rx! * Radian!: Ry! = Ry! * Radian!: Rz! = Rz! * Radian!
   ' Z rot, cw
   X1! = X! * Cz! + Y! * Sz!: Y1! = X! * -Sz! + Y! * Cz!: Z1! = Z!
   ' Y rot, back
   X2! = X1!: Y2! = Y1! * CX! + Z1! * -Sx!: Z2! = Y1! * Sx! + Z1! * CX!
   ' X rot, left
   SUB42X! = X2! * CY! + Z2! * -Sy!: SUB42Y! = Y2!: Sub42Z! = X2! * Sy! + Z2! *
CY!
End Sub
```

VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the Claims:

Claim 1 has been amended as follows:

1. (Amended) I claim a method comprising of a technology element that calculates Multi-Axes Tool Compensation internal to a central mathematical set of algorithms in memory of the CNC controller which ties all of the provided set of commands together as described ~~in the steps and elements of which comprise:~~ and shown in the provided flowchart in block diagram form FIG 10 enumerated as Paragraph [0030] titled as Intelligent Database subroutine and Database subroutine which calls, ties to and works together with the group of elements titled the collection of mathematical subroutine elements enumerated as Paragraphs [0031]-[0054] and specifically linked to and shown in FIG 10 of the block diagram as it interacts with the Element titled DbAtr enumerated as paragraph [0043], Element titled DbGet enumerated as paragraph [0044], Element titled DbSet enumerated as paragraph [0045] and Element titled DbSetAtrCur enumerated as paragraph [0046]. To further recite all of the elements, components and steps completely constituting every aspect we further list and explain the technology elements as comprising of:

a. The user setting his or her preferences for the values or amounts to compensate into boxes on an operator screen, such as the example screen in FIG 1. for the boxes labeled tool size, horizontal offset, vertical offset, tool wear, corner radius, bottom angle, side angle and tool length. These interact with the G code program and other values optionally inputted or gathered as variables when the math calculations are performed.

   The user input boxes on FIG 1 specifically are read and stored by the Subroutine Element Form_Load enumerated as [0031] which reads in all data from user input boxes from FIG 1 and stores them into the Database Element as described and enumerated as paragraph [0030]. Further database variables and user settings both public, global and private call, ties to and works together with the database element titled subroutine DbAtr enumerated as [0043] which is a Database element to store geometry properties, error, conditions and positions. This ties to and works with the Element titled Intelligent Database subroutine enumerated as paragraph [0030] and the database element titled Subroutine DbGet enumerated as [0044] which gets Database item coordinate, property and position from random file which works with the Element titled Intelligent Database subroutine enumerated as paragraph [0030] the database element titled Subroutine DbSet enumerated as [0045]. The DbSet Database element sets item coordinate, property and position from random file. The database element titled Subroutine GloRead enumerated as [0032] reads in all global and public data from user input boxes plus any proprietary settings from FIG 1 and stores them into the Database Element as described in and enumerated as paragraph [0030] titled as Intelligent Database subroutine and Database subroutine.

b. The user must repeat the steps in Claim 1.a setting and entering his or her preferences for each tool description. There is no limit to the number of tools, machine types or tool shape combinations to enter.

c. An industry standard G Code program, as in FIG 9, containing tool positions based on non-compensated original part geometry data, interact with the Multi-Axes tool compensation calculations when they are applied. These are the original tool positions that the user supplies in which the calculations are applied. These interact with values provided on the tool parameter screen. For each multi-axes X,Y,Z,A,B,C value entered in the G Code program, the controller will calculate a compensated value based on the amounts entered into the tool parameter screen as in the example screen in FIG 1.

d. A set of optional text entered commands are provided to interact and be directly entered onto the operator screen to override or toggle features on/off and adjust values:

TOOLCOMP OFF	'Turns all compensation off.
TOOLCOMP LEFT	'Comps tool in 2D to the left.
TOOLCOMP RIGHT	'Comps tool in 2D to the right.
TOOLCOMP 3DCOMP	'3D comp based on vector and gouge parameter.
TOOLCOMP 3DADJUSTZ	'3D comp lifts Z axis only but keeps X,Y.
TOOLCOMP 3DOFFSET	'3D parallel offset only - based on vector and 'no gouge parameter.
TOOLCOMP 5AXIS	'5-axis comp based on vector and gouge parameter.
TOOLCOMP LLIMIT45	'Give angle which will specify a gouge to omit tool.

e. A multi-axes tool positioner in a tool holder mounted to a machine's spindle cuts the part as shown in FIG 7 and FIG 8.

f. The process of gathering the user-entered information, preferences, values, amounts, on/off options on the operator screen as in FIG 1. or as entered by optional text commands as in Claim 1.d interact with the original tool positions as provided in the G code program, as in FIG 9, to provide the mathematical variables when processed by a set of described central mathematical routines internal to the CNC Controller as outlined in the DETAILED DESCRIPTION OF THE INVENTION section. The various methods for gathering the information are incidental as to how the central set of math routines that perform these calculations receive them.

Claim 2 has been amended as follows:

2. (Amended) I claim a method for Multi-Axes Tool Compensation ~~element according to Claim 9,~~ specifically pointing out by distinctly claiming the subject matter as what I regard as my invention is an element, which automatically calculates tool gouge avoidance protection internal to the CNC controller's central set of math routine algorithms as shown: ~~in FIG 5 Dim "E" Item 7.~~

a. Elements and components are depicted in FIG 3 Item 5, FIG 4 DIM "D" Item 6, FIG 5 DIM "E" Item 7 and FIG 6 DIM "R" Item 8.

b. Relationships between the Tool Parameter, Tool Definitions, Machine and Fixture offset elements and how they work together are shown by flowchart in block diagram form.

c. We further recite the relationships as outlined in the DETAILED DESCRIPTION OF THE INVENTION in paragraphs enumerated as [0024] through [0028] as the means in which the necessary parameters are gathered from the user and stored using computer variables within the computer's memory as shown in paragraph enumerated as [0030] as the element titled Database subroutine then passed to the subroutines of the technology element as depicted in paragraphs [0022] as the element titled Vector and Matrix subroutine, [0023] as the element titled Gouge subroutine and [0029] as the element titled Central subroutine.

Claim 3 has been amended as follows:

3. (Amended) I claim a method ~~pertaining to Claim 1~~ for Multi-Axes Tool Compensation, specifically pointing out by distinctly claiming the subject matter as what I regard as my invention is an element which automatically contains algorithms to lift the tool to safe positions or skip the move when necessary by determining if the LLIMIT parameter, as shown in FIG 5 Dim "E" Item 7, is in violation of any surrounding obstacles as determined by a user-defined variable value as enter on the operator screen in FIG 1.

a. This claim is a ~~an alternative~~ claim method of calculating tool gouge avoidance and tool protection as outlined in Claim 2.

b. As the key objective we use a method of user-definable command elements that override, replace and redefine the variables gathered from the user and stored using computer variables within the computer's memory as shown in paragraph enumerated as [0030] as the element titled Database subroutine.

c. We list and recite the elements of the user-definable commands as shown in paragraphs enumerated as [0015] through [0021] of the DETAILED DESCRIPTION OF THE INVENTION as:

TOOLCOMP OFF
TOOLCOMP LEFT
TOOLCOMP RIGHT
TOOLCOMP 3DCOMP
TOOLCOMP 3DADJUSTZ
TOOLCOMP 3DOFFSET
TOOLCOMP 5AXIS
TOOLCOMP LLIMIT45

which are further recited to show the relationships of how the user-definable command set overrides, replaces and redefines the user input as outlined in FIG 1 on the operator screen and described in the DETAILED DESCRIPTION OF THE INVENTION in paragraphs enumerated as [0001] through [0012].

Claim 4 has been amended as follows:

4. (Amended) I claim a method ~~pertaining to Claim 1 which does not depend on the CNC programmer to re-define~~ specifically pointing out by distinctly claiming the subject matter as what I regard as my invention is an element to redefine, replace and override the tool position coordinates when the tool characteristics change:

a. Relationships between the user-definable command method elements and how they work together are shown in the flowchart in block diagram form, as in FIG 10.

b. As the key objective we use a method of user definable command elements that over ride, replace and re-define the variables gathered from the user and stored using computer variables within the computers memory as shown in paragraph enumerated as [0030] as the element titled Database subroutine.

We list and recite the elements of the user definable commands as shown in paragraphs enumerated as [0015] through [0021] of the DETAILED DESCRIPTION OF THE INVENTION as:

TOOLCOMP OFF
TOOLCOMP LEFT
TOOLCOMP RIGHT
TOOLCOMP 3DCOMP
TOOLCOMP 3DADJUSTZ
TOOLCOMP 3DOFFSET

TOOLCOMP 5AXIS
TOOLCOMP LLIMIT45

which are further recited to show the relationships of how the user-definable command set overrides, replaces and redefines the user input as outlined in FIG 1 on the operator screen and described in the DETAILED DESCRIPTION OF THE INVENTION in paragraphs enumerated as [0001] through [0012]

Claim 5 has been canceled.

Claim 6 has been canceled.

Claim 7 has been canceled.

Claim 8 has been canceled.

Claim 9 has been amended as follows:

9. (Amended) I claim an algorithm element ~~according to Claim 1 which does not store or pass the compensated positions by geometry alone but rather~~ expands the intelligence of each calculation for compensated tool positions based on an artificial intelligence algorithm element.

a. The artificial intelligence algorithm element is actually a live, real-time, ever-changing database in the machine's memory that remembers by learning from what the machine can and cannot do. The database is a storage of events, variables as an internal list of conditions and positions kept in standard random access memory as outlined by the various variables used by the central set of math algorithms.

b. Specifically pointing out by distinctly claiming the subject matter as what I regard as my invention is an element for which we recite as a method to store events, conditions, positions and errors into computer variables within the computer's memory as shown in paragraph enumerated as [0030] as the element titled Database subroutine.

c. Relationships between the intelligent database element that stores the events, conditions, positions and errors into computer variables and how they work together are shown in the flowchart in block diagram form, as in FIG 10.

<u>In the Drawings:</u>

The content of FIG 1 has been amended as indicated by the red marking within the drawing itself. The page numbers for FIG 1 through FIG 9 have been amended, as indicated by the red markings, as a result of the addition of FIG 10.

Invention Title: Multi-Axes Tool Compensation – 3D and 5-axis real-time interactive tool compensation inside the CNC machine tool controller.
Inventor: Gary John Corey
Application No. 10/079,309
Inventor's Phone No.: (951) 674-8100

1/10

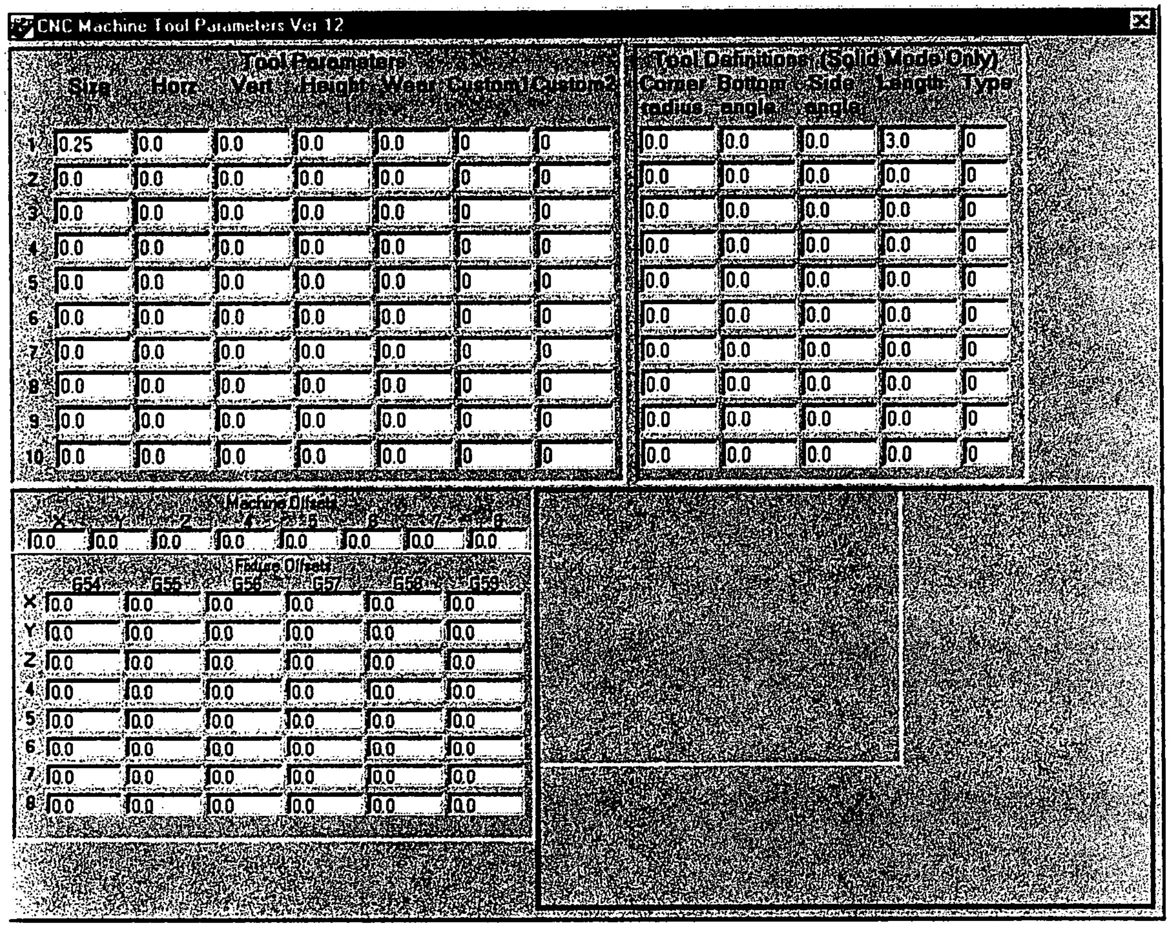

FIG 1.

Invention Title: Multi-Axes Tool Compensation – 3D and 5-axis real-time interactive tool compensation inside the CNC machine tool controller.

Inventor: Gary John Corey
Application No. 10/079,309
Inventor's Phone No.: (951) 674-8100

2/10

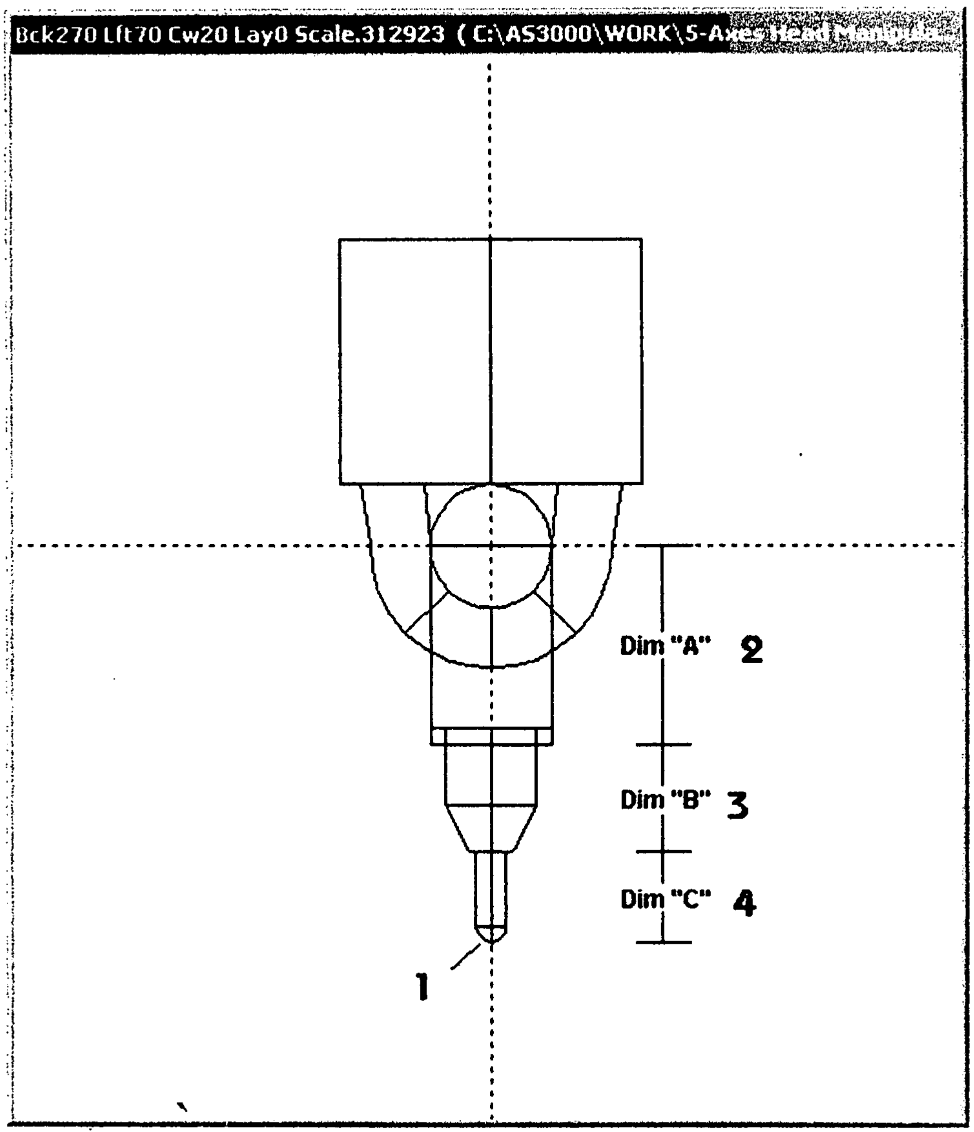

FIG 2.

Invention Title: Multi-Axes Tool Compensation – 3D and 5-axis real-time interactive tool compensation inside the CNC machine tool controller.
Inventor: Gary John Corey
Application No. 10/079,309
Inventor's Phone No.: (951) 674-8100

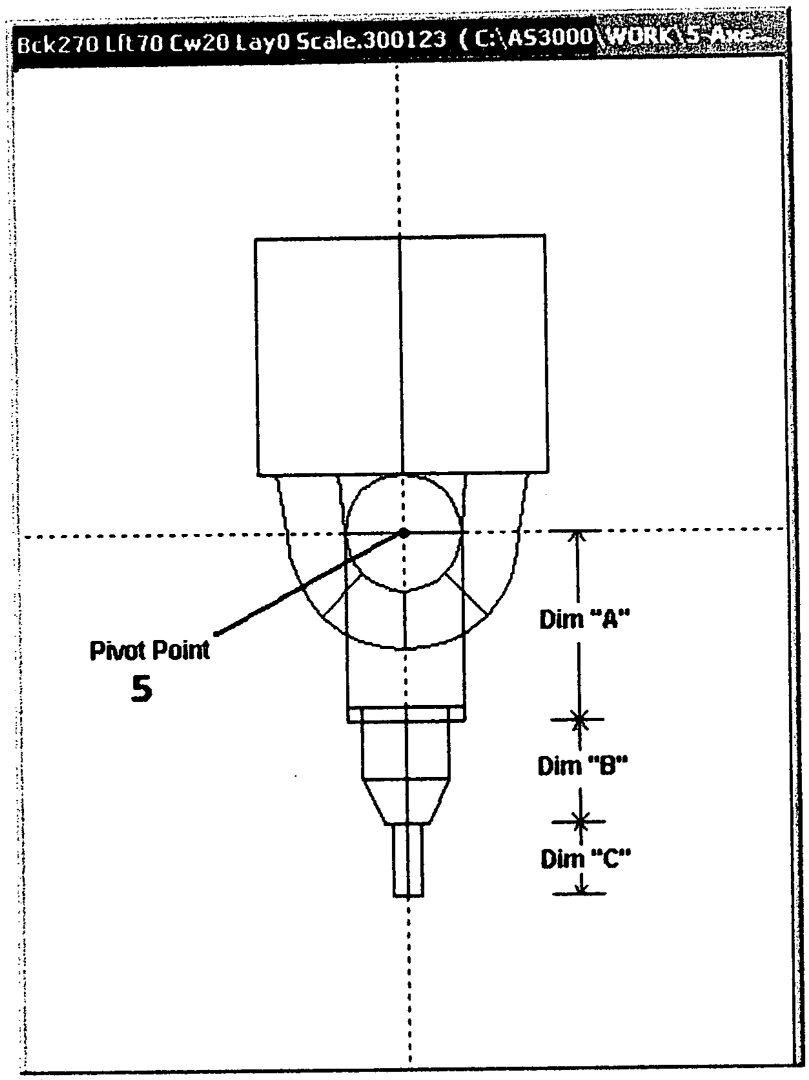

FIG 3.

Invention Title: Multi-Axes Tool Compensation – 3D and 5-axis real-time interactive tool compensation inside the CNC machine tool controller.
Inventor: Gary John Corey
Application No. 10/079,309
Inventor's Phone No.: (951) 674-8100

4/10

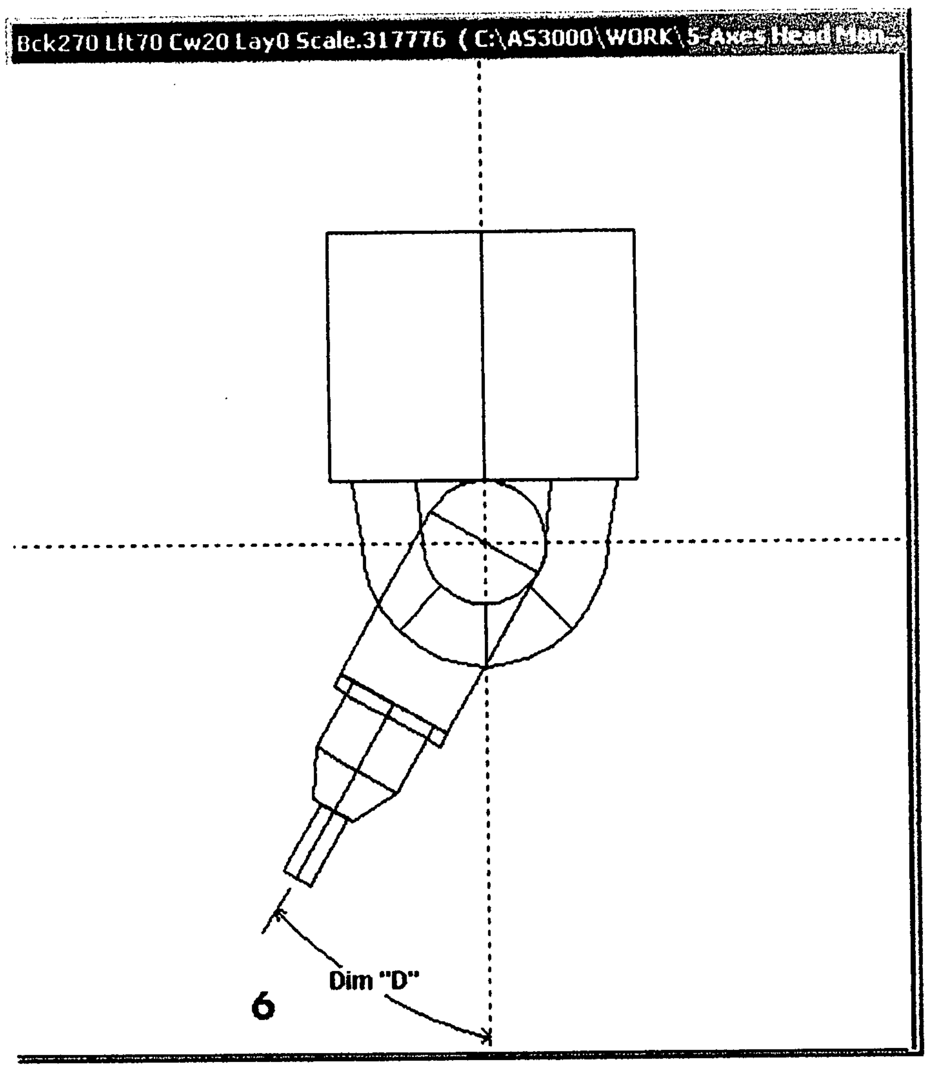

FIG 4.

Invention Title: Multi-Axes Tool Compensation – 3D and 5-axis real-time interactive tool compensation inside the CNC machine tool controller.

Inventor: Gary John Corey
Application No. 10/079,309
Inventor's Phone No.: (951) 674-8100

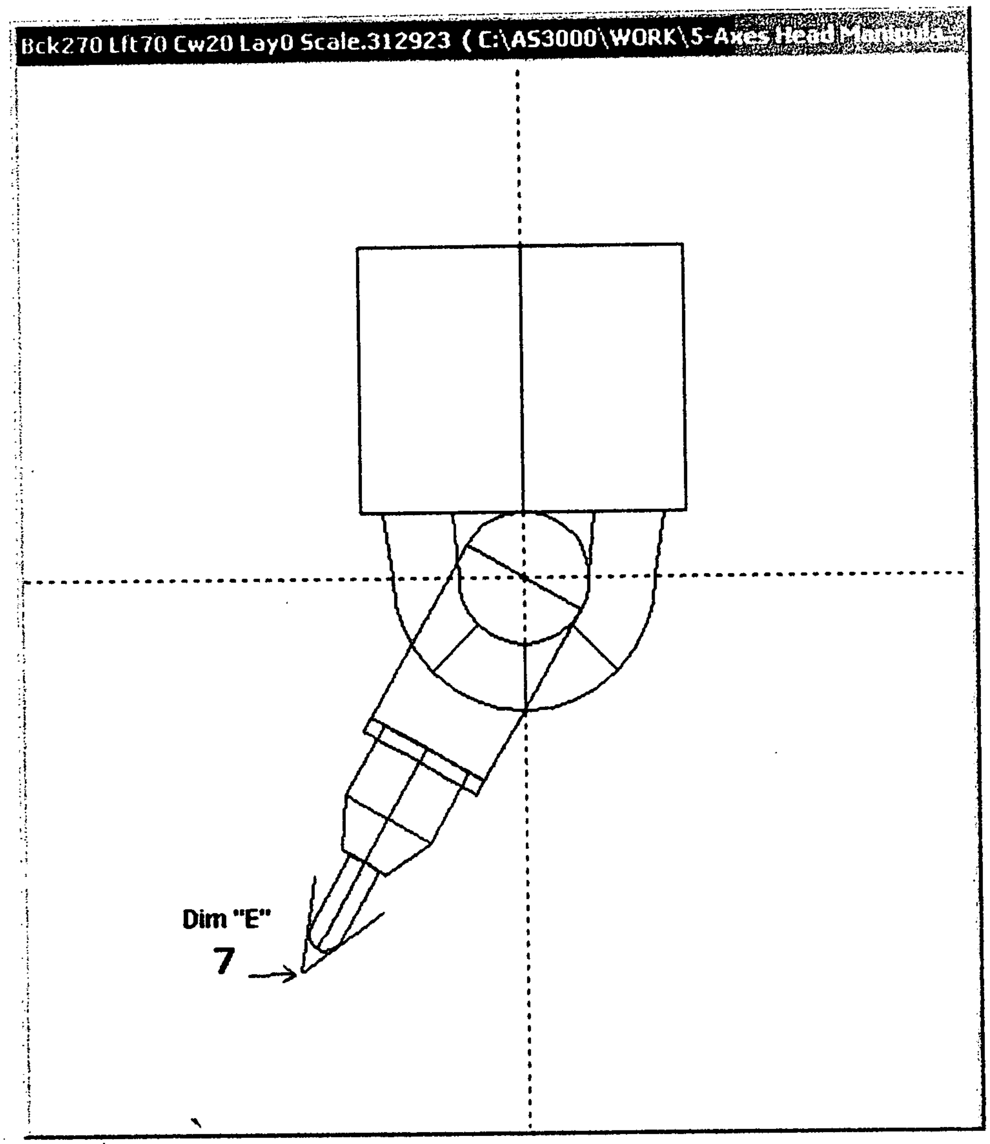

FIG 5.

Invention Title: Multi-Axes Tool Compensation – 3D and 5-axis real-time interactive tool compensation inside the CNC machine tool controller.
Inventor: Gary John Corey
Application No. 10/079,309
Inventor's Phone No.: (951) 674-8100

6/10

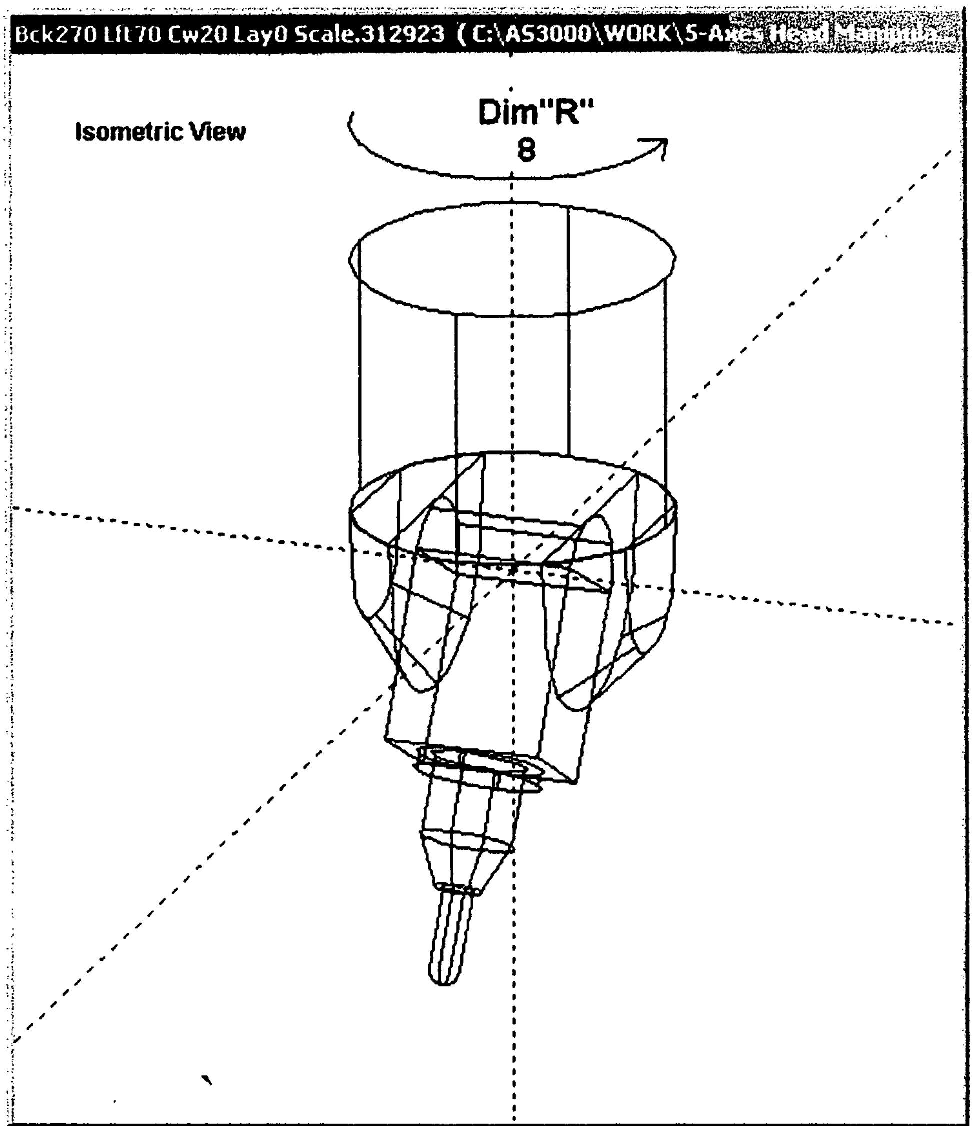

FIG 6.

Invention Title: Multi-Axes Tool Compensation – 3D and 5-axis real-time interactive tool compensation inside the CNC machine tool controller.
Inventor: Gary John Corey
Application No. 10/079,309
Inventor's Phone No.: (951) 674-8100

7/10

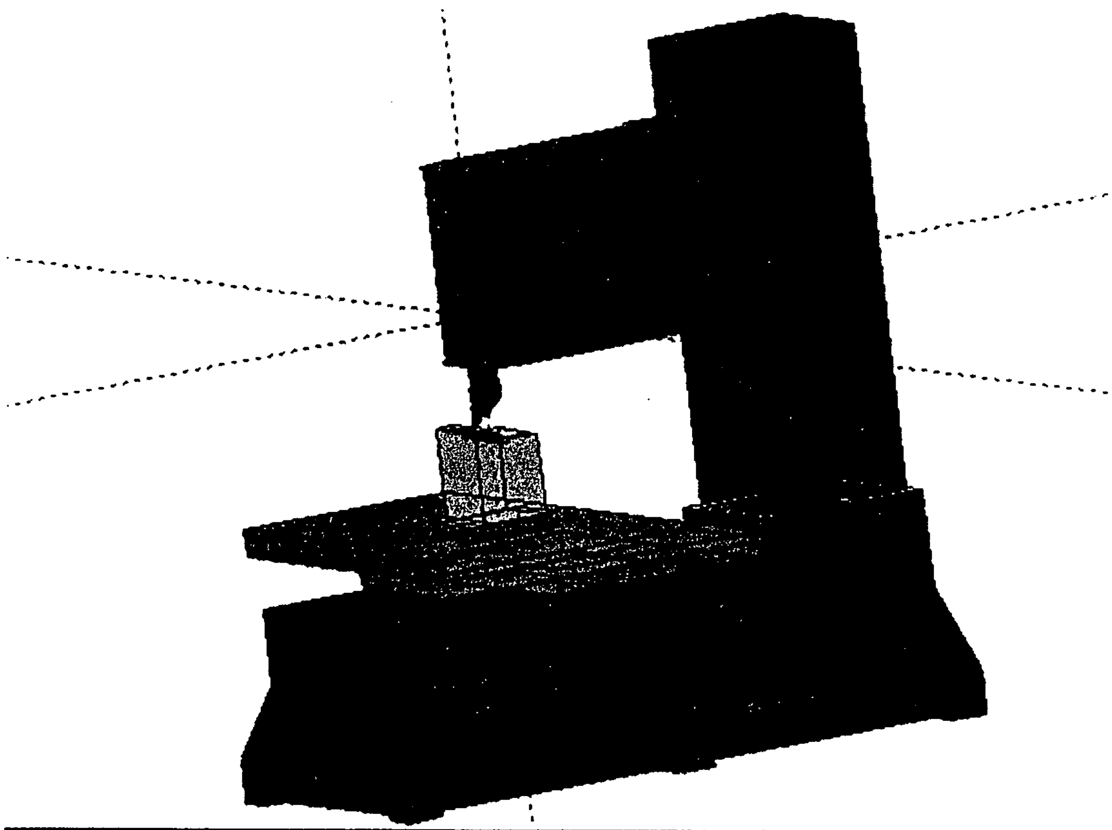

FIG 7.

Invention Title: Multi-Axes Tool Compensation – 3D and 5-axis real-time interactive tool compensation inside the CNC machine tool controller.
Inventor: Gary John Corey
Application No. 10/079,309
Inventor's Phone No.: (951) 674-8100

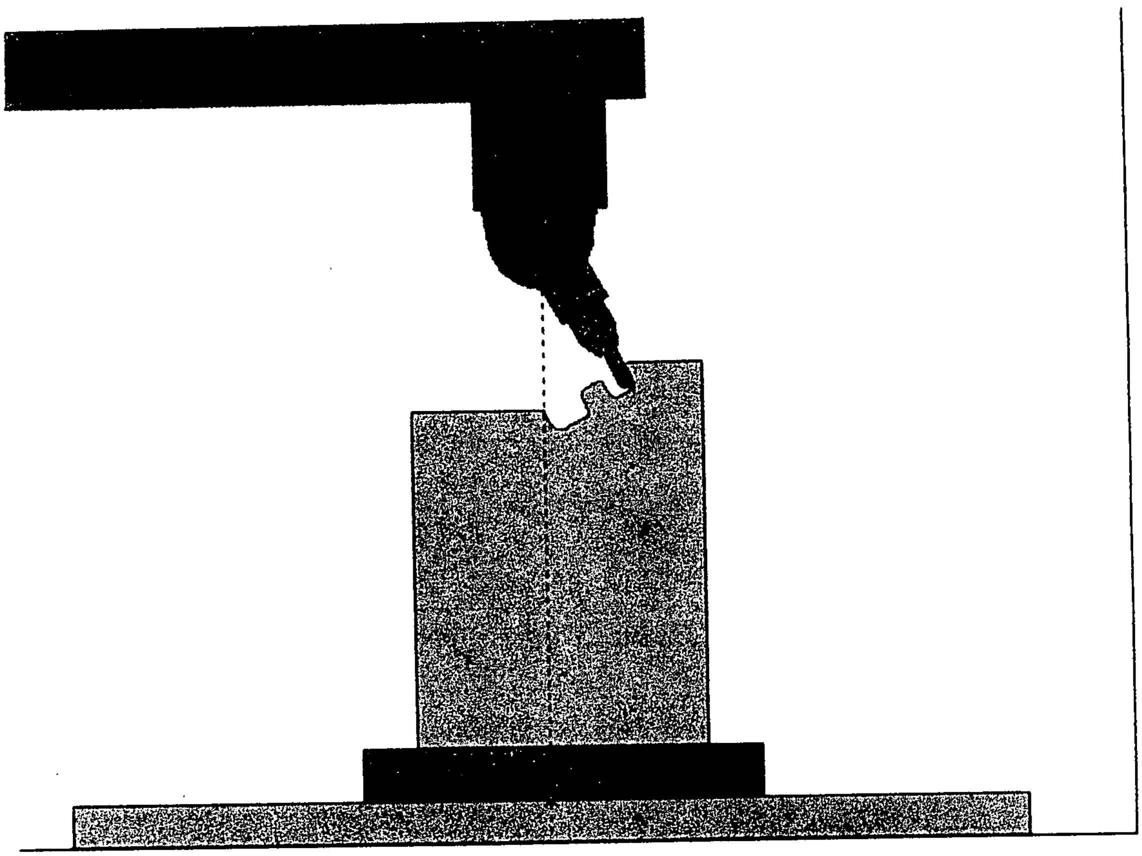

FIG 8.

Invention Title: Multi-Axes Tool Compensation – 3D and 5-axis real-time interactive tool compensation inside the CNC machine tool controller.
Inventor: Gary John Corey
Application No. 10/079,309
Inventor's Phone No.: (951) 674-8100


```
%
N10 T01 M6
N20 G90 S200 M3
N30 G0 A270. B0
N40 X0 Y-21. Z0 M8
N50 Z20.5
N60 G1 Y-10.933 Z17.9365 A270. B-30. F10.
N70 Y-3.2465 Z10.75 A270. B-60.
N80 Y0 Z.5 A270. B-90.
N90 Y-3.2465 Z10.75 A90. B-60.
N100 Y-10.933 Z17.9365 A90. B-30.
N110 Y-21. Z20.5 A90. B0
N120 G1 Z0
N130 G0 A0 B0
N140 X0 Y-21. Z0 S200 F10.
N150 G1 Z20.5
N160 G1 Y-10.9332 Z17.9367 A0 B-29.9993 F10.
N170 Y-3.2463 Z10.7498 A0 B-60.0007
N180 Y0 Z.5 A0 B-90.
N190 Y-3.2465 Z10.75 A180. B-60.
N200 Y-10.933 Z17.9365 A180. B-30.
N210 Y-21. Z20.5 A180. B0
N220 Z0
N230 G0 A0 B0
N240 M30
%
```

FIG 9.

DRAWINGS

FIG 10.: Flowchart in block diagram form of each technology element.

Invention Title: Multi-Axes Tool Compensation - 3D and 5-axis real-time interactive tool compensation inside the CNC machine tool controller.
Inventor: Gary John Corey
Application No. 10/079,309
Inventor's Phone No.: (951) 674-8100

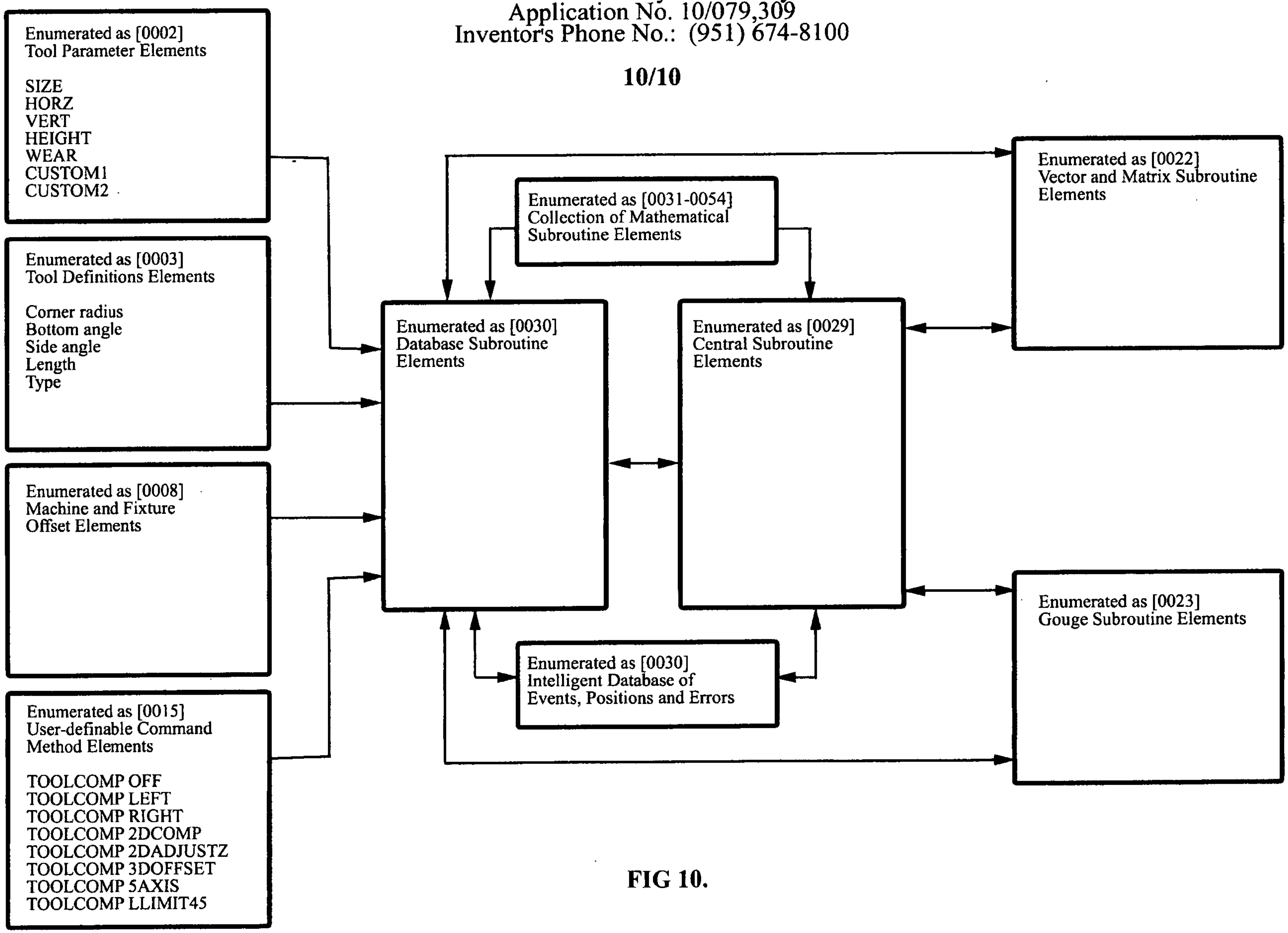

FIG 10.